2002年日本オペレーションズ・リサーチ学会
春季研究発表会

最小全域木を全列挙するアルゴリズム*

加入申請中　防衛大学校情報工学科　渡辺 宏太郎† WATANABE Kohtaro
01700900　防衛大学校情報工学科　山田 武夫　YAMADA Takeo
01107880　防衛大学校情報工学科　片岡 靖詞　KATAOKA Seiji

1　はじめに

最小全域木問題はグラフ理論の基本問題として，これまで数多くの研究がなされてきた [1]．これらのほとんどは与えられたグラフの最小全域木のひとつを見出すことを目的としている．また全域木をすべて列挙するアルゴリズムも活発に研究されてきた [2]．これに対して，本稿ではグラフ中の最小全域木をすべて列挙する問題を考察する．対象とするグラフは V，E を節点及び枝の集合とする無向グラフ $G=(V,E)$ で，各枝には整数値のコスト $c: E \to Z_+$ が付されているものとする．また，$n := |V|$, $m := |E|$ とし，$z = z(T)$ で G の全域木 T のコストを表す．

2　分割統治法

G における最小全域木 T の 1 つは Kruskal，Prim などのアルゴリズムによって容易に求めることができるが，そのコストを $\underline{z}$ と記す．

G の枝集合で，サイクルを含まないものを $F \subseteq E$，これと素な枝集合を $R \subseteq E$ とする．全域木 T が (F,R)-許容であるとは，T が F をすべて含み，R は含まないことを言う．F は固定枝，R は禁止枝の集合を表す．部分問題 $P(F,R)$ を，(F,R)-許容な全域木で，コストが $\underline{z}$ であるものをすべて列挙する問題とすると，元問題は $P(\emptyset,\emptyset)$ となる．

(F,R)-許容な最小全域木をひとつだけ見つけることは（Kruskal などの方法を適当に修正して）容易であるが，このアルゴリズムを An_MST(F,R) とすると，$P(F,R)$ を解くアルゴリズムを右上のように構成することが出来る．

図 1 は All_MSTs(F,R) の挙動を表す．太線は固定枝，= は禁止枝で，破線（+太線）は最小全域木である．このように，部分問題は木構造をなし，葉以外の内点ではそれぞれ異なる最小全域木が 1 つずつ見つかって，全体として 6 個の最小全域木が求められている．

アルゴリズム All_MSTs(F,R)

Step 1: $T :=$ An_MST(F,R).
Step 2: $T=\emptyset$ または $z(T) > \underline{z}$ならば return.
Step 3: Tの枝集合を $F \cup \{e^u, \cdots, e^{n-1}\}$ とし，
$i = u, \ldots, n-1$ に対して以下を実行する．
(i) $F' := F \cup \{e^u, \ldots, e^{i-1}\}, R' := R \cup \{e^i\}$.
(ii) All_MSTs(F',R') を再帰呼び出し．

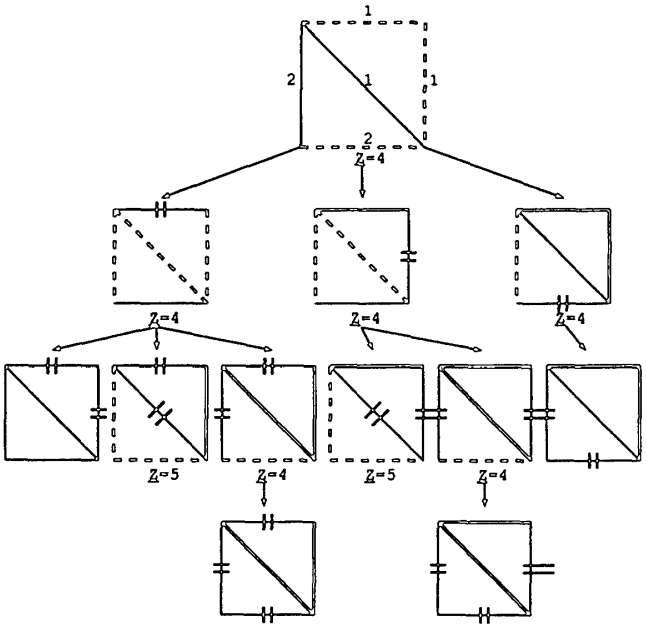

図 1: アルゴリズムの挙動

3　時間及び空間計算量の評価

G に含まれる最小全域木の総数を N とする．1 つの部分問題から派生する子問題の個数は最大 $n-1$ 個なので，全体で生成される部分問題の個数は高々 $Nn+1$ となる．G で最小全域木を求めるのに要する計算時間を $T_{\mathrm{MST}}(n,m)$ とすると，各部分問題を処理する時間もこれ以下なので，全体としての計算量は $(Nn+1)T_{\mathrm{MST}}(n,m)$ となる．

*富山国際会議場 大手町フォーラム, H14.3.27-28
†E-mail: wata@nda.ac.jp

次に, All_MSTs(F,R) は再帰型のアルゴリズムで, その深さは最大限 m であり, 各部分問題で新たに固定または禁止される枝を記憶しておく必要があるので, 必要メモリサイズは $O(m)$ である. 以上より, 次を得る

定理 1 All_MSTs$(\emptyset,\emptyset)$ の時間計算量は $O((Nn+1)T_{\mathrm{MST}}(n,m))$, 空間計算量は $O(m)$ である.

上で, $T_{\mathrm{MST}}(n,m)$ は Kruskal 法の標準的な実現では $T_{\mathrm{MST}}(n,m)=m\log n$ と評価される.

4 数値実験

上のアルゴリズムを C 言語で実現し, IBM RS/6000 Model 270 上で実験を行った.

4.1 動作確認

完全グラフ K_n で, 枝の荷重をすべて 1 とした場合, 全域木はすべて最小全域木で, その総数は既知である. 表 1 に, いくつかの例について, 全域木総数 (N) と生成子問題数 (#sub), CPU 時間を計測した結果を示す. 全域木総数はすべて Kirchhoff の式から求められる数と一致している.

表 1. K_nにおける全域木の全列挙

例題	N	#sub	CPU 秒
K_6	1296	2117	0.02
K_7	16807	26830	0.20
K_8	262144	412015	3.49
K_9	4782969	7433032	70.90
K_{10}	100000000	154076201	1645.73

4.2 完全グラフ

次に, 完全グラフ K_n で, 枝荷重を [1,1000] の乱数とした場合の結果を表 2 に示す. 各行は, 独立な 10 回の試行の平均値で, それぞれ最小全域木の荷重 ($\underline{z}$), 最小全域木の数 (N), 生成された子問題数 (#sub) と CPU 時間を表す.

また, 枝荷重の範囲を [1,100] とした場合の結果を表 3 に示す. これらの表から, 問題がある程度以上の規模となるにつれ, また枝荷重のばらつきが小さくなるにつれ, 最小全域木の数が急激に増えて計算が困難になることがわかる.

4.3 平面グラフ

$P_{n.m}$ を節点数 n, 枝数 m の平面グラフで, 900×700 の領域に描画されているものとする. 枝荷重は Euclid 距離である. これについての結果を表 4 に示す. やはりある程度以上のサイズになると, 最小全域木の数が急激に増加している.

表 2. 実験結果 (K_n, 荷重は [1,1000] でランダム)

例題	$\underline{z}$	N	#sub	CPU 秒
K_{20}	1114.2	1.1	20.2	0.02
K_{40}	1224.0	2.5	59.9	0.06
K_{60}	1260.5	4.1	101.8	0.13
K_{80}	1253.0	3.1	136.9	0.20
K_{100}	1264.1	9.3	227.4	0.42
K_{120}	1251.6	14.3	361.4	0.85
K_{140}	1281.3	123.2	1768.9	5.00
K_{160}	1274.3	337.4	7966.7	29.51
K_{180}	1288.8	3980.0	63892.2	315.82
K_{200}	1296.4	7434.4	145946.9	938.66

表 3. 実験結果 (K_n, 荷重は [1,100] でランダム)

例題	$\underline{z}$	N	#sub	CPU 秒
K_{20}	120.9	3.6	38.2	0.04
K_{40}	141.1	11.0	112.8	0.12
K_{60}	152.5	1669.2	8840.1	10.33
K_{80}	165.9	1494553.0	4647056.0	6234.56

表 4. 実験結果 (平面グラフ, 荷重は Euclid 距離)

例題	$\underline{z}$	N	#sub	CPU 秒
$P_{20,46}$	1718	1	20	0.01
$P_{50,127}$	3376	1	50	0.03
$P_{100,260}$	4591	4	124	0.15
$P_{200,560}$	6618	32	1650	4.45
$P_{400,1120}$	9301	24576	1523587	9736.97

5 むすび

最小全域木を全列挙するアルゴリズムを提示した. 数値実験のさらなる詳細は当日報告したい.

参考文献

[1] T.L. Magnanti *et al*, "Optimal Trees", Hand Books in Operations Research and Management Science, Vol.7, Elsevier, (1995), 503-616.

[2] Y. Matsui, *et al.*, Algorithm for Combinatorial Enumeration Problem, http://dmawww.epfl.ch/roso.mosaic/kf/enum/comb/combenum.html.